Naval Research Laboratory

Washington, DC 20375-5320

NRL/MR/6410--95-7797

Simulation of a Torpedo Launch Using A 3-D Incompressible Finite Element Solver and Adaptive Remeshing

RAVI RAMAMURTI
WILLIAM SANDBERG

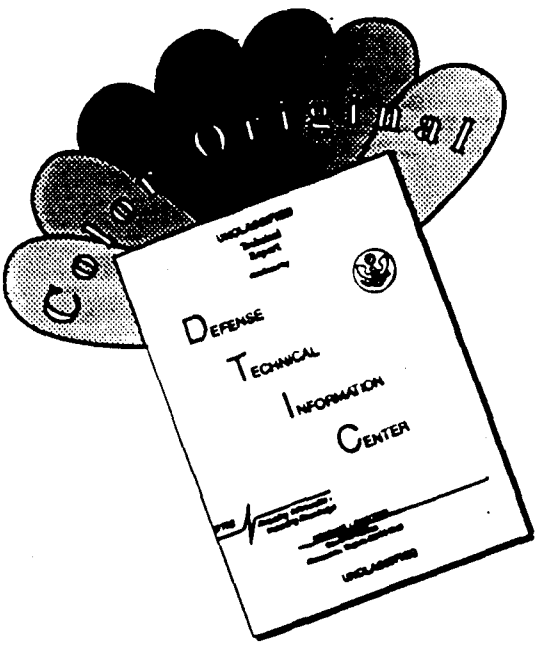
Center for Reactive Flow and Dynamical Systems
Laboratory for Computational Physics and Fluid Dynamics

RAINALD LÖHNER

Institute for Computational Sciences and Informatics
The George Mason University, Fairfax, VA 22030

November 30, 1995

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF COLOR PAGES WHICH DO NOT REPRODUCE LEGIBLY ON BLACK AND WHITE MICROFICHE.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave Blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	November 30, 1995	NRL Memorandum Report

4. TITLE AND SUBTITLE

Simulation of a Torpedo Launch Using A 3-D Incompressible Finite Element Solver and Adaptive Remeshing

5. FUNDING NUMBERS

PE — 62326N

6. AUTHOR(S)

Ravi Ramamurti, William C. Sandberg, and Rainald Löhner*

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/MR/6180--95-7795

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Naval Undersea Warfare Center,
Newport Division
1176 Howell Street
Newport, RI 02841-5047

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

*Institute for Computational Sciences and Informatics, The George Mason University, Fairfax, VA 22030

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

An implicit finite element solver for three dimensional incompressible flows with adaptive remeshing has been developed. This flow solver is employed for the simulation of an unsteady torpedo launch from a submarine. Several mesh movement algorithms have been developed and implemented. The use of gliding points on surfaces in the vicinity of a moving body has proven valuable in reducing the number of remeshings and thus reducing the total CPU time required for the simulation. The adaptive flow solver has been parallelized and is written in a manner that allows portability among various supercomputer architectures.

14. SUBJECT TERMS

Torpedo launch
Unstructured meshes
Incompressible flows
Adaptive remeshing

15. NUMBER OF PAGES

33

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std 239-18
298-102

CONTENTS

Simulation of a Torpedo Launch Using a 3-D Incompressible Finite Element Solver and Adaptive Remeshing

1. INTRODUCTION

The flow field generated during a torpedo launch from a submarine is quite complex. The complexities arise due to the outer submarine flow entering the cavity of the torpedo bay in the initial stages of the launch, the induced flow field due to the relative motion of the bodies present, and, last but not least, the effect of the water jet that is employed to propel the weapon.

In the past, two methods have been employed to predict the torpedo flow field [1]. The first method is based on a slender body and crossflow drag theories to predict the linear and non-linear hydrodynamics of the torpedo body when it emerges in the launchway, in free flight through, and beyond the submarine flow field. Different hydrodynamic equations and flow field assumptions are used in the three different flow regimes. The submarine flow field has historically been computed in this approach using potential flow panel codes which do not account for the local effects due to the shutter cavity geometry, the torpedo body, or the ejection jet. The second method is based on steady, incompressible, potential flow assumption for the entire flow field. The torpedo is assumed to be on the tube's firing centerline. The hydrodynamic coefficients obtained based on these assumptions, which are used to impose forces and moments on the vehicle, are applied even as the vehicle deviates from the centerline. This could result in inaccuracies in prediction of the launch trajectory while the torpedo is in the launchway. The method being developed in the on-going task couples an incompressible flow solver together with adaptive remeshing thus making the use of hydrodynamic coefficients unnecessary. Forces and moments can be obtained directly from the computed pressure field for the instantaneous dynamic conditions. This information can then be provided to the potential flow trajectory code. The main goal of this effort is to perform this coupling in an efficient manner.

Advances in computer speed and memory offer the potential of simulating these flows in a timely manner. Thus, one can expect this type of advanced CFD to gradually take the lead role in the design process for applications such as this. The present effort represents a first step in this direction. First we state the goals and design considerations for the present effort. Our goal is to develop an efficient tool for the simulation of incompressible flows about complex geometry two-body flows. This brings with it the following requirements:

a) Ability to grid complex geometries rapidly: this requirement is met by using **unstructured grids** (tetrahedra) in conjunction with CAD-tools to define the geometries.

b) Ability to mesh appropriately high Reynolds-number cases: this requirement is met by employing arbitrary semi-structured grids close to wetted surfaces and wakes [2].

Manuscript approved November 3, 1995.

c) Simple elements: in order to be as fast as possible, the overhead in building element matrices, residual vectors, etc. should be kept to a minimum. This requirement is met by employing **simple, low-order elements that have all the variables (velocities, pressure) at the same location.**

d) Ability to simulate transient and steady-state solutions: in many cases, particular high Reynolds-number flows, the resulting flow fields will exhibit a tendency to unsteadiness. It is important that the flow solver be able to capture the unsteadiness of a flow field if such exists. The present flow solver is built as time-accurate from the onset, allowing local timestepping as an option.

e) Inviscid, Incompressible Flows: in order to study trends, an engineer may wish to approximate a high Reynolds-number flow by an inviscid flow. An inviscid flow will require a much coarser grid, which translates into fast turnaround. Therefore, the flow solver must also have the **ability to solve the incompressible Euler equations.**

f) Body Motion: a further degree of complexity is added for the class of problems considered here due to the relative motion of the bodies present. The bodies move through an already complex, highly nonlinear flow field, modifying it constantly.

2. THE INCOMPRESSIBLE FLOW SOLVER

The incompressible Navier-Stokes equations in the Arbitrary Lagrangian Eulerian (ALE) form can be written as

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}_a \cdot \nabla \mathbf{v} + \nabla p = \nabla \cdot \sigma \ , \tag{1a}$$

$$\nabla \cdot \mathbf{v} = 0 \ , \tag{1b}$$

where p denotes the pressure, $\mathbf{v}_a = \mathbf{v} - \mathbf{w}$ the advective velocity vector (flow velocity $\mathbf{v}$ minus mesh velocity $\mathbf{w}$) and both the pressure p and the stress-tensor σ have been normalized by the (constant) density ρ, are discretized in time using an implicit timestepping procedure. The resulting expressions are subsequently discretized in space using a Galerkin procedure using linear tetrahedral elements with all variables located at the nodes (u,v,w,p) [3]. This leads to the following matrix system

$$\begin{bmatrix} \mathbf{K_{vv}} & \mathbf{K_{vp}} \\ \mathbf{K_{pv}} & \mathbf{K_{pp}} \end{bmatrix} \begin{pmatrix} \Delta \mathbf{v} \\ \Delta \mathbf{p} \end{pmatrix} = \begin{pmatrix} \mathbf{r_v} \\ \mathbf{r_p} \end{pmatrix} . \tag{2}$$

For practical 3-D problems, the direct solution of this coupled system of equations is beyond the reach of current supercomputer technology. The most common way to reduce the storage and CPU-requirements is to decouple the system of equations into separate elliptic problems in the following iterative loop, sometimes referred to as **preconditioned Uzawa algorithm** [3]:

U1) Given $\Delta \mathbf{p}$, compute $\Delta \mathbf{v}$:

$$\mathbf{K_{vv}}\Delta\mathbf{v} = \mathbf{r_v} - \mathbf{K_{vp}}\Delta\mathbf{p} \tag{3a}$$

U2) Given $\Delta\mathbf{v}$, compute $\Delta\mathbf{p}$:

$$\left(-\mathbf{K_{pv}}\tilde{\mathbf{K}}_{\mathbf{vv}}^{-1}\mathbf{K_{vp}} + \mathbf{K_{pp}}\right)\Delta\mathbf{p} = \mathbf{r_p} - \mathbf{K_{pv}}\tilde{\mathbf{K}}_{\mathbf{vv}}^{-1}\mathbf{r_v} \tag{3b}$$

U3) If not yet converged: goto U1).

Of course, little is gained unless an inexpensive approximation $\tilde{\mathbf{K}}_{\mathbf{vv}}^{-1}$ to $\mathbf{K_{vv}}^{-1}$ can be found. Fortunately, such an approximation exists if we simply take

$$\tilde{\mathbf{K}}_{\mathbf{vv}}^{-1} = \mathbf{M_l}^{-1} , \tag{4}$$

where $\mathbf{M_l}$ denotes the lumped mass-matrix. The reason why this approximation works is that in the limit, as the mesh size h tends to zero, Eqs.(3b,4) will approximate the analytic result

$$\nabla^2 p = -\nabla \cdot \mathbf{v}_a \cdot \nabla \mathbf{v} , \tag{5}$$

which is obtained by taking the divergence of the momentum equations (1a).

As one can see, the bulk of the computational work lies in the Poisson-like problems given by Eqs.(3a,3b). These matrix systems are solved iteratively using a preconditioned gradient algorithm (PCG). The preconditioning is achieved through linelets [4]. The flow solver has been successfully evaluated for both 2-D and 3-D, laminar and turbulent flow problems by Ramamurti *et al.* and was also parallelized in order to improve its efficiency and portability to various supercomputer architectures [5,6].

3. RIGID BODY MOTION

In order to fully couple the motion of rigid bodies with the hydrodynamic or aerodynamic forces exerted on them, consistent rigid body motion integrators must be developed.

The governing equations of motion for rigid bodies can be found in textbooks on classical mechanics. The rigid body motion in 3-D is not straightforward due to its nonlinear character [7]. Therefore, a more detailed description of the equations and the incorporation of the rigid body motion in the numerical scheme for solving the fluid flow are described in this section.

Given the position vector of any point of the body

$$\mathbf{r} = \mathbf{r}_c + \mathbf{r}_0 , \tag{6}$$

where $\mathbf{r}_c$ is the position vector of the center of mass and $\mathbf{r}_0$ is the position vector of any point relative to the center of mass, then the velocity and acceleration of this point can be written as

$$\dot{\mathbf{r}} = \dot{\mathbf{r}}_c + \dot{\mathbf{r}}_0 = \mathbf{v}_c + \boldsymbol{\omega} \times \mathbf{r}_0 \quad , \tag{7a}$$

and

$$\ddot{\mathbf{r}} = \dot{\mathbf{v}}_c + \dot{\boldsymbol{\omega}} \times \mathbf{r}_0 + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{r}_0) \tag{7b}$$

respectively.

The linear and angular accelerations $\dot{\mathbf{v}}_c$ and $\dot{\boldsymbol{\omega}}$ can be related to the fluid pressure through the balance of forces and moments which are written as follows.

$$m\dot{\mathbf{v}}_c = \sum \mathbf{F} = m\mathbf{g} - \int_\Gamma p\mathbf{n}d\Gamma \tag{8a}$$

$$\begin{aligned} \boldsymbol{\Theta}\dot{\boldsymbol{\omega}} + \int_\Omega (\boldsymbol{\omega} \cdot \mathbf{r}_0) \cdot (\mathbf{r}_0 \times \boldsymbol{\omega}) d\Omega &= \sum \mathbf{r}_0 \times \mathbf{F} \\ &= -\int_\Gamma p\mathbf{r}_0 \times \mathbf{n}d\Gamma \end{aligned} \tag{8b}$$

where m is the mass of the rigid body, p is the hydrodynamic pressure and

$$\begin{aligned} \boldsymbol{\Theta} &= tr(\mathbf{I}) \cdot \mathbf{1} - \mathbf{I} \\ &= \begin{Bmatrix} I_{yy} + I_{zz} \, , & -I_{xy} \, , & -I_{xz} \, , \\ -I_{xy} \, , & I_{xx} + I_{zz} \, , & -I_{yz} \, , \\ -I_{xz} \, , & -I_{yz} & I_{xx} + I_{yy} \end{Bmatrix} \quad , \end{aligned} \tag{9}$$

is the matrix of moment of inertia. Observe that in 2-D, the second term on the left-hand side disappears, considerably simplifying the equations. However, in 3-D it usually does not. Another complication that arises only in 3-D is the temporal variation of the inertial matrix $\boldsymbol{\Theta}$. As one can see from Eq.(9), the values of $\boldsymbol{\Theta}$ will vary as the body rotates. This implies that during the simulation one has to follow the local frame of reference of the body.

The velocity and position of the rigid bodies in time are updated using an explicit time-marching scheme. Thus, the velocity $\mathbf{v}_c$ is updated as

$$\mathbf{v}_c^{n+1} = \mathbf{v}_c^n + \Delta t \dot{\mathbf{v}}_c^n \quad . \tag{9}$$

The magnitude of the timestep Δt is unknown before the start of the flow field update. In the present study, the timestep corresponding to the previous flow field update is used instead. This

implies that the body movement is lagging behind the flow field by at most one timestep. However, the actual error is much smaller as the magnitude of Δt does not change abruptly. The velocity at the body surface is then updated using the average velocity of the center of mass

$$\mathbf{v}_c^{avg} = 0.5\left(\mathbf{v}_c^{n+1} + \mathbf{v}_c^{n}\right) \tag{10}$$

together with Eq. (7a).

4. ADAPTIVE REMESHING

Fast regridding capabilities are needed because the motion of bodies may be severe, leading to distorted elements which in turn lead to poor numerical results. As the bodies in the flow field may undergo arbitrary movement, a fixed mesh structure will lead to badly distorted elements. This means that at least a partial regeneration of the computational domain is required. On the other hand, as the bodies move through the flow field, the positions of relevant flow features will change. Therefore, in most of the computational domain a new mesh distribution will be required. The idea is to regenerate the whole computational domain adaptively, taking into consideration the current flow field solution. In order to generate or regenerate a mesh we use the advancing front technique [8,9].

4.1 Recent Developments

A typical simulation where bodies undergo severe motion typically requires hundreds of remeshings. Therefore, the grid generator must be reliable and fast.

4.1.1 Reliability:

We have recently increased the reliability of the grid generator to a point where it can be applied on a routine basis in a production environment. This significant increase in reliability was achieved by:

a) not allowing any unacceptable elements during the generation process; and

b) enlarging and remeshing those regions where new elements could not be introduced.

Thus, we first attempt to complete the mesh, skipping those faces that do not give rise to good elements. If pockets of unmeshed regions remain, we enlarge them somewhat, and regrid them. This 'sweep and retry' technique has proven extremely robust and reliable. It has also made smoothing of meshes possible: if elements with negative or small Jacobians appear during smoothing, these elements are removed. The unmeshed regions of space are then regridded. By being able to smooth, the mesh quality was improved substantially.

4.1.2 Speed:

The following means are used to achieve speed:

a) Use of optimal data structures: In order to perform the required search operations as quickly as possible, the following data structures are used:

- Heap-lists to find the next face to be deleted from the front;
- Quad-trees (2-D) and Octrees (3-D) to locate points that are close to any given location;
- Linked lists to determine which faces are adjacent to a point.

The detailed implementation of these data-structures may be found in [10].

b) Filtering: Typically, the number of close points and faces is far too conservative, i.e. large. As an example, consider the search for close points: there may be up to eight points inside an octant, but of these only one may be close to the face to be taken out. The idea is to filter out these 'distant' faces and points in order to avoid extra work afterwards. While the search operations are difficult to vectorize, these filtering operations lend themselves to vectorization in a straightforward way, leading to a considerable overall reduction in CPU requirements.

c) Automatic Reduction of Unused Points: As the front advances into the domain and more and more tetrahedra are generated, the number of tree-levels increases. This automatically implies an increase in CPU-time, as more steps are required to reach the lower levels of the trees. In order to reduce this CPU-increase as much as possible, all trees are automatically restructured. All points which are completely surrounded by tetrahedra are eliminated from the trees. We have found this procedure to be extremely effective. It reduced the asymptotic complexity of the grid generator to less than $O(N \log N)$. In fact, in most practical cases one observes a linear $O(N)$ asymptotic complexity, as CPU is traded between subroutine call overheads and less close faces on average for large problems.

d) Global h-refinement: While the basic advancing front algorithm is a scalar algorithm, h-refinement can be completely vectorized. Therefore, the adaptive remeshing process can be made considerably faster by first generating a coarser, but stretched mesh, and then refining globally this first mesh with classic h-refinement [11]. Typical speed-ups achieved by using this approach are 1:6 to 1:7.

Currently, the advancing front algorithm constructs grids at a rate of 60,000 for very small grids to 80,000 tetrahedra per minute for large grids on the CRAY-C90.

4.2 Local Remeshing

Practical simulations revealed that the appearance of badly distorted elements occurred more often than expected. Given the relatively high cost of global remeshing, we explored the idea of local remeshing in the vicinity of the elements that became too distorted. An outline of the method developed for local remeshing is as follows:

L.1 Identify the badly distorted elements in the layers that move, writing them into a list `LEREM(1:NEREM)`.

L.2 Add to this list the elements surrounding these badly distorted elements.

L.3 Form 'holes' in the present mesh by:

L.3.1 Forming a new background mesh with the elements stored in the list `LEREM` .

L.3.2 Deleting the elements stored in `LEREM` from the current mesh.

L.3.3 Removing all unused points from the grid thus obtained.

L.4 Recompute the error indicators and new element distribution for the background grid.

L.5 Regrid the 'holes' using the advancing front method.

Typically, only a very small number of elements (< 10) becomes so distorted that a remeshing is required. Thus, local remeshing is a very economical tool that has allowed us to reduce CPU-requirements by more than 60% for typical runs.

5. MESH MOVEMENT ALGORITHMS

An important question from the point of view of mesh distortion and remeshing requirements is the algorithm employed to move the mesh. Given that the mesh velocity on the moving surfaces of the computational domain is prescribed,

$$\mathbf{w}|_{\Gamma_m} = \mathbf{w}_0 \ , \tag{11}$$

and, at a certain distance from these moving surfaces, as well as all the remaining surfaces the mesh velocity vanishes

$$\mathbf{w}|_{\Gamma_0} = 0 \ , \tag{12}$$

the question is how to obtain a mesh velocity field $\mathbf{w}$ in such a way that element distortion is minimized. A number of algorithms have been proposed. They may be grouped together into the following categories.

a) prescribing the mesh velocity analytically,

b) smoothing the coordinates, and

c) smoothing the velocity field.

5.1 Prescription Via Analytic Functions

In this case the mesh velocity is prescribed to be an analytic function based on the distance from the surface. Using heap-lists, as well as other optimal data structures, the distance from the surface may be obtained in $O(N \log N)$ operations, where N is the number of grid points. Given this distance ρ, and the point on the surface closest to it $\mathbf{x}|_\Gamma$, the mesh velocity is given by

$$\mathbf{w} = \mathbf{w}(\mathbf{x}|_\Gamma) \cdot f(\rho) \ . \tag{13}$$

The function $f(\rho)$ assumes the value of unity for $\rho = 0$, and decays to zero as ρ increases. This makes the procedure somewhat restrictive for general use, particularly if several moving bodies are present in the flow field. On the other hand, the procedure is extremely fast if the initial distance ρ can be employed for all times [12].

5.2 Smoothing of the Coordinates

In this case, we start with the prescribed boundary velocities. This yields a new set of boundary coordinates at the new timestep.

$$\mathbf{x}^{n+1}\Big|_\Gamma = \mathbf{x}^n|_\Gamma + \Delta t \cdot \mathbf{w}|_\Gamma \quad . \tag{14}$$

Based on these new values for the coordinates of the boundary points, the mesh is smoothed. In most cases to date, a simple spring analogy smoother has been employed. The new values for the coordinates are obtained iteratively via a relaxation or conjugate gradient scheme [13,14]. As before, a good initial guess may be extrapolated via

$$\mathbf{x}_0^{n+1} = 2\mathbf{x}^n - \mathbf{x}^{n-1} \quad . \tag{15}$$

The smoothed mesh velocity is then given by

$$\mathbf{w} = \frac{1}{\Delta t} \cdot \left(\mathbf{x}^{n+1} - \mathbf{x}^n \right) \quad . \tag{16}$$

Most of the potential problems that may occur for this type of mesh velocity smoothing are due to initial grids that have not been smoothed. For such cases, the velocity of the moving boundaries is superposed to a fictitious mesh smoothing velocity which may be quite large during the initial stages of a run. Moreover, if spring analogy smoothers are employed, there is no guarantee that negative elements won't appear.

5.3 Smoothing of the Velocity Field

In this case, the mesh velocity is smoothed, based on the exterior boundary conditions given by Eqs.(11,12). The aim, as stated before, is to obtain a mesh velocity field $\mathbf{w}$ in such a way that element distortion is minimized. Consider for the moment the 1-D situation sketched in Fig. 1. At the left end of the domain, the mesh velocity is prescribed to the w_0. At the right end, the mesh velocity vanishes. If the mesh velocity decreases linearly, i.e.

$$\frac{\partial w}{\partial x} = g_w \quad , \tag{17}$$

then the elements will maintain their initial size ratios. This is because for any two elements, the change in size δh during one timestep is given by

$$\delta h = (w_2 - w_1) \cdot \Delta t = \Delta w \cdot \Delta t \quad . \tag{18}$$

The size-ratio of any two elements i, j is given by

$$\begin{aligned} \left.\frac{h_i}{h_j}\right|^{n+1} &= \frac{h_i + \Delta w_i \cdot \Delta t}{h_j + \Delta w_j \cdot \Delta t} \\ &= \frac{h_i + g_w \cdot h_i \cdot \Delta t}{h_j + g_w \cdot h_j \cdot \Delta t} = \left.\frac{h_i}{h_j}\right|^{n} \quad . \end{aligned} \tag{19}$$

This implies that all elements in the regions where mesh velocity is present will be deformed in roughly the same way (think of 3-D situations with rotating or tumbling bodies immersed in the flow field). Solutions with constant gradients are reminiscent of Laplacian operators, and indeed, for the general case, the mesh velocity is obtained by solving

$$\nabla \cdot \mathbf{k} \cdot \nabla \mathbf{w} = 0 \quad , \tag{20}$$

with the Dirichlet boundary conditions given by Eqs.(11,12). This system is again discretized using Finite Elements. The resulting system of equations may be solved via relaxation as

$$C^{ii} \Delta \mathbf{w}^i = -\Delta \tau \cdot K^{ij} \left(\mathbf{w}^i - \mathbf{w}^j \right) \quad , \tag{21}$$

where

$$C^{ii} = \sum_{i \neq j} |K^{ij}| \tag{22}$$

and the optimal Δt-sequence is given by

$$\Delta \tau^i = \frac{1}{1 + cos[\frac{\pi(i-1)}{n_v}]} \quad , i = 1, n \quad . \tag{23}$$

The starting estimate for new mesh velocity vector may be extrapolated from previous timesteps, e.g.

$$\mathbf{w}_0^{n+1} = 2\mathbf{w}^n - \mathbf{w}^{n-1} \quad . \tag{24}$$

Since the mesh movement is linked to a timestepping algorithm for the fluid part, and since the body movement occurs at a slow pace compared to the other wavespeeds in the coupled fluid/solid system, normally no more than two to five steps are required to smooth the velocity field sufficiently,

i.e. $3 \leq n \leq 5$. The overhead incurred by this type of smoothing is very small compared to the overall costs for any ALE-type methodology for Euler or Navier-Stokes flow solvers.

If the diffusion coefficient appearing in Eq.(20) is set to $k = 1$, a true Laplacian velocity smoothing is obtained. This yields the most 'uniform deformation' of elements, and therefore minimizes the number of remeshings or remappings required. Alternatively, for element-based codes, one may approximate the Laplacian coefficients K^{ij} in Eq.(21) by

$$\nabla^2 \mathbf{w} \approx -(\mathbf{M}_l - \mathbf{M}_c) \cdot \mathbf{w} \quad . \tag{25}$$

This approximation is considerably faster for element-based codes (for edge-based codes there is no difference in speed between the true Laplacian or this expression), but it is equivalent to a diffusion coefficient $k = h^2$. This implies that the gradient of the mesh velocity field will be larger for smaller elements. These will therefore distort at a faster rate than the larger elements. Obviously, for uniform grids this is not a problem. However, for most applications the smallest elements are close to the surfaces that move, prompting many remeshings.

Based on the previous arguments, one may also consider a diffusion coefficient of the form $k = h^{-p}, p > 0$. In this case, the gradient of the mesh velocity field will be larger for the larger elements. The larger elements will therefore distort at a faster rate than the smaller ones, a desirable feature for many applications.

One of the possible disadvantages of mesh velocity smoothing is that the appearance of negative elements, *i.e.,* inverted volumes, cannot be guaranteed. While there are a number of situations where the appearance of such elements cannot be avoided (e.g. tumbling or rotating objects), a number of researchers have preferred to work with coordinate smoothing. In order to see more clearly the relative merits of the mesh velocity smoothing and the coordinate smoothing, consider the simple box shown in Fig. 2. Suppose that face A is being moved in x-direction. For the case of mesh velocity smoothing, only displacements in the x-direction will appear. This is because the Dirichlet boundary conditions given by Eqs.(11,12) do not allow any mesh velocity other than in the x-direction to appear. On the other hand, for the case of coordinate smoothing, there will, in all likelihood appear mesh velocities in the y and z-directions. This is because, as the mesh moves, the smoothing technique will result in displacements of points in the y and z-directions, and hence velocities in the y and z-directions. Therefore, this will not retain the original representation of the surface of body. In order to recover the original surface definition, one has to use the CAD definition to correct the points on the body surface. This operation has to be performed every time step and will be costly in terms of CPU time.

6. REGION OF MOVING ELEMENTS

As the elements (or edges) move, their geometric parameters (shape-function derivatives, jacobians, etc.) need to be recomputed every timestep. If the whole mesh is assumed to be in motion, then these geometric parameters need to be recomputed globally. In order to save CPU-time, only a small number of elements surrounding the bodies are actually moved. The remainder of the field is then treated in the usual Eulerian frame of reference, avoiding the need to recompute geometric parameters. This may be accomplished in a variety of ways, of which the two most common are

a) by identifying several layers of elements surrounding the surfaces that move, and b) by moving all elements within a certain distance from the surfaces that move. Both approaches have their advantages and disadvantages, and are therefore treated in more detail.

a) Layers of Moving Elements: In this case the elements moved are obtained by starting from the moving surfaces, and performing n passes over nearest neighbours to construct the n layers of elements that move. This procedure is extremely fast and works only with integer variables. On the other hand, for situations where the element size varies rapidly, the extent of the moving mesh region can assume bizarre shapes. This, in turn, may force many remeshings at a later stage. This type of procedure is most commonly used for Euler calculations [15-17].

b) Elements Within a Distance: This second approach requires the knowledge of the distance a point has from the moving surfaces. All elements within a prescribed distance from the moving surfaces are considered as moving. Although this procedure required more CPU-time when being built, it offers the advantage of a very smooth boundary of the moving mesh region. Moreover, by specifying two distances, the region close to the moving surfaces may be moved in the same way the surfaces move, while further away the mesh velocity is smoothed as before. This allows the movement of Navier-Stokes type grids that are very elongated, and hence sensitive to any kind of distortion [5].

The extent of the region of moving elements chosen, given by the number of layers and/or the distance from the moving surfaces, can have a pronounced effect on the overall cost of a simulation. As the number of elements moved increases, the time-interval between regridding increases, but so also does the cost per timestep. Therefore, one has to strike a balance between the CPU requirements per timestep and the CPU requirements per regridding. In most of the calculations carried out by the authors, it was found that five to twenty layers of elements represent a good compromise.

6.1 Layers of Gliding Points

In the case of the torpedo launch simulation, the gap between the launchway tube and the torpedo surface is quite small. This allows only a very few, (typically one or two,) layers of elements in this gap. Due to the closeness of the moving body to the stationary tube and to the lack of sufficient layers of elements, the distortion of the mesh will be pronounced, resulting in increased number of remeshings. One way to circumvent this problem is to allow the points on the surface of the launchway tube to glide along the surface. The velocity of these gliding points are set from the velocity of moving bodies. This is achieved by first smoothing the velocity over the layers of elements that surround the moving body. Then, the velocity at points along the gliding surfaces are smoothed. The velocity along the gliding surfaces are then restricted to be tangential to the surface. This ensures that the surface does not get distorted. During a 20 time step simulation of the torpedo launch, the number of local remeshings was reduced from 5 to 1 and the number of global remeshings was reduced from 2 to 1 by invoking gliding points along the launchway tube surface.

7. THE OVERALL ALGORITHM

Before discussing the application to the torpedo launch problem, we will summarize the coupled algorithm of the flow solver with adaptive remeshing and rigid body motion as follows:

O.1 Advance the solution one timestep.

- A.1 Compute the body forces and moments from the pressure field and any exterior forces.
- A.2 Taking into consideration the kinematic constraints for the body movements, update the velocities of the bodies at $t = t^{n+1}$: $\mathbf{v}_c^{n+1}, \boldsymbol{\omega}^{n+1}$. At the same time, obtain the average velocities $\mathbf{v}_c^{av}, \boldsymbol{\omega}^{av}$ for the time-interval $[t^n, t^{n+1}]$.
- A.3 With the average velocities $\mathbf{v}_c^{av}, \boldsymbol{\omega}^{av}$, obtain the velocities $\mathbf{w}_\Gamma$ on the surface of each body for the time-interval $[t^n, t^{n+1}]$.
- A.4 Given the surface velocities $\mathbf{w}_\Gamma$ on the boundaries of the global domain, obtain the global velocity field $\mathbf{w}_\Omega$ for the element movement.
- A.5 Advance the solution by one timestep using the flow solver. This yields the actual timestep Δt^n.
- A.6 Given the actual timestep Δt^n and the velocity field for the element movement $\mathbf{w}_\Omega$, update the coordinates of the points.
- A.7 Update the shape-function derivatives and other geometric parameters for the elements that have been moved.
- A.8 Update the centers of mass $\mathbf{r}_c$ for the bodies, as well as the coordinates of the points defining the body geometry.

O.2 If the grid has become too distorted close to the moving bodies: adaptively remesh these regions.

O.3 If the desired number of timesteps between global remeshings has elapsed: adaptively remesh the complete computational domain.

O.4 If the desired time-interval has elapsed: Stop. Otherwise, advance the solution further (go to O.1)

8. DISCUSSION OF RESULTS

8.1 Torpedo Launch

One of the goals of this effort is to couple the adaptive remeshing flow solver, FEFLOIC, with the trajectory prediction code UVLDS. The schematic of this hybrid simulation process is shown in Fig. 3. The simulation starts with the UVLDS code and passes information about the torpedo positions and angles to FEFLOIC. The flow solver then advances the flow with the prescribed motion of the torpedo from the current position to the UVLDS predicted positions. The pressure on the torpedo is then integrated and the hydrodynamic forces and moments are then passed on to the UVLDS code. This completes the loop for the hybrid simulation. During the flow solver phase, numerous remeshings will be required in order to advance the flow from the current body position to the final position. In order for this hybrid simulation to be an efficient prediction tool,

the number of remeshings and hence the CPU time required by FEFLOIC have to be minimized. Hence, a major portion of this effort is directed towards minimizing the number of remeshings and the total CPU time required by the flow solver.

The inviscid flow past a submarine and a torpedo during the launch process was simulated first. The grid that was employed consisted of approximately 60,000 points and 280,000 tetrahedra. The motion of the torpedo was prescribed from experiments and is shown in Fig. 4. First a steady state solution of the flow was obtained. In this case, the torpedo was stationary in the launchway with the shutterway fully opened. The unsteady simulation was continued with this steady state as the starting point. The starting time for this unsteady simulation, $t_1 = 0.42$ secs., was obtained from the experimental trajectory corresponding to the position of the torpedo in the launchway. The velocity of the torpedo and the velocity of the water that is employed to propel the torpedo were obtained from experiments and imposed as boundary conditions. The launch simulation was continued until a time $t_2 = 0.94$ secs. During this interval, there were 765 local remeshings and 122 global remeshings. The CPU time for advancing the flow by one timestep was approximately 16 seconds. The CPU times for one local and one global remeshings were 36.0 secs. and 293.0 secs. respectively. The force on the torpedo was computed by integrating the surface pressure (Eq. 8a). The acceleration of the torpedo along the centerline of the launchway tube was then computed and compared to the experimental acceleration. This solution was dominated by the transients generated during the impulsive starting process.

In order to eliminate these transients, a new initial flow configuration was developed. This was done by including a portion of the torpedo which would glide and expand from the base of the launchway tube in a manner similar to an extrusion process as shown in Fig. 5. The points on this cylindrical portion were allowed to glide. The velocity at these points was obtained using velocity smoothing as explained in the previous section. The launch simulation was started from time $t_0 = 0.0$ secs. The process was continued until the extruded body length becomes equal to the torpedo length. At this instant, $t = 0.635$ secs., the torpedo completely clears the launchway base. Hence, a new configuration containing the actual torpedo geometry replaced the extruded body. In order to perform this metamorphosis, the flow solution from the extruded body had to be interpolated to this new configuration. The inflow condition at the base of the launchway tube was also modified to include the water jet that is used to propel the torpedo. The launch simulation was continued until t_2 in 2100 time steps. During the interval $t_1 - t_2$, there were 386 local remeshings and 53 global remeshings. The reduction in the number of remeshings was due to the use of gliding points. These reductions translated to a savings of 10 CPU hours for this portion of the simulation. Further, the local and global remeshing algorithms in addition to the flow solver, were parallelized on the Cray C-90 using auto-tasking. For steady state calculations, *i.e.,* without remeshing, the speed-up obtained was 3.2 out of 4 processors. For unsteady flow simulation with global remeshing, the speed-up obtained in a production environment (not dedicated), was 1.7 out of 4 processors. This reduced the wall clock time by a factor of 1.7.

The time history of pressure at three stations along the inner wall near the shutterway opening were tracked and are shown in Fig. 6. At the station farthest from the opening, the pressure reached a peak at $t = 0.375$ secs. and dropped to a minimum at $t = 0.605$ secs. The pressure reached a peak again at $t = 0.655$ secs. This corresponds to the pressure increase due to the water jet at the base reaching this location. At the station midway between the opening and the first station, the pressure reaches a maximum at $t = 0.655$ secs. and a minimum at $t = 0.705$ secs. At the

station on the opening of the shutterway, the pressure reaches a maximum at $t = 0.735$ secs. and a minimum at $t = 0.78$ secs. The minimum pressure at these stations occurs when the nose of the torpedo had just crossed that location. The time history of forces on the torpedo, shown in Fig. 7, exhibits a peak at $t = 0.3$ secs. corresponding to the peak acceleration in the experiment at around this time. Also, there is a large shift in both the forces and moments at $t = 0.635$ secs. which corresponds to the time at which both the configuration change and the velocity boundary condition at the launchway base were imposed. Figure 8a shows the surface triangulation at t_2 as the torpedo emerges from the shutterway. Figures 8b and 8c show the surface pressure and velocity vectors at this instant.

Forces and moments were computed on the extruded portion of the torpedo which extends beyond the launchway opening up to $t = 0.635$ secs. and on the complete torpedo after this instant of time. The computed centerline acceleration of the torpedo, $a_{extruded}$ is shown in Fig. 9. In order to compare the centerline acceleration of the torpedo with experiment, the force due to the water pressure at the base of the torpedo has to be included. This force was obtained from experiments, and the corrected acceleration is a_{corr} in Fig. 9. This exhibits a double peak which was absent in the experiments. The most likely reason for the discrepancy are lack of grid resolution and the leakage of water at the base of the launchway. These issues are being investigated further.

In order to assess grid dependence of the solution and to study the diffusive effects of remeshings, a finer grid was employed in the vicinity of the torpedo. The generated grid for the initial configuration consists of 155K points and 819K tetrahedra approximately and had two layers of elements in the narrow gap between the torpedo and the launchway tube. The computed acceleration is compared to the coarse grid results and is shown in Fig. 10. From this figure, it is clear that both $a_{extruded}$ and a_{corr} remain nearly unchanged due to grid refinement. Therefore, the coarse grid is sufficient for capturing the dynamics due to the surface pressure.

Another major deviation of the simulation from the experiment is the leakage of water that is used to propel the weapon at the base of the launchway. In order to address this issue, the boundary conditions at the base of the launchway on the extruded portion of the torpedo were corrected. This was done so that the velocity at these points was equal to the water jet velocity. The unsteady simulation was performed until $t = 0.39$ secs. In this interval, 5 global remeshings and 59 local remeshings were required. The force on the torpedo was computed and the centerline acceleration was compared to the previous computation and the experiment and is shown in Fig. 11. The peak acceleration without water leakage was approximately $280 ft/sec^2$ compared to the experimental value of $200 ft/sec^2$. The effect of including the water leakage is to cut the peak acceleration to about $220 ft/sec^2$.

One of issues examined was whether the computed force due to pressure includes the force due to added mass. The force experienced by a submerged body moving at a constant acceleration a in a potential flow is equal to $(m + m_{add})a$, where m is the mass of the body and m_{add} is the added mass. This m_{add} depends on the shape of the body and the direction and type of acceleration the body is subjected to. In order to clarify the nature of the added mass contribution to our results, the force on a 1:6 prolate spheroid moving at a constant acceleration was computed. First, a steady state solution was obtained. Then, the body was accelerated with a prescribed constant acceleration. During this computation there were several local and global remeshings. The force during the interval $t = 0.0$ to $t = 1.0$, was increasing steadily and not reaching an asymptotic value. If the computation has to be continued for a longer period, there is the danger of the body

exiting the computational domain. A simple and obvious fix to this problem is to linearly increase the inflow velocity of the computational domain. For this case, the computation was continued to $t = 12.0$. Even in this case, the force did not reach an asymptotic value. Another test case where the inflow velocity was accelerated up to $t = 0.5 Secs.$ and held constant thereafter, was computed. The time history of the force is shown in Fig. 12. For this case, the flow reached a steady state and the total force on the body was negligible and similar to the steady force on the body before acceleration. It is now becoming apparent that extraction of the added mass contribution is not a straightforward matter. Our computations are solutions to the Euler equations and thus include rotational effects which are not present in a potential flow solution. Also, the added mass is a local effect and our computations are not local but include the increase in kinetic energy throughout the domain. It is therefore considered not worthwhile to attempt to isolate the added mass contribution to the Euler computations.

8.2 Integration with UVLDS

For the integration of the FEFLOIC code with the UVLDS code of NUWC, first, the flow solver was made into a subroutine which can be called from the UVLDS code. The flow solver passes the forces computed on the torpedo to the trajectory code and receives the new position information. A coarse grid consisting of approximately 7500 points was generated to test the integration. First, a steady state solution was obtained on the SGI workstation at NUWC. The integrated flow solver advanced the flow for 80 time steps and experienced difficulties after that. These were traced to differences in the coordinate systems employed by the two codes and they were corrected. Also, the restart capability in FEFLOIC was modified so that the two codes were not synchronized.

8.3 Turbulence Models

For viscous flow simulations of the launch problem, it is important to have an efficient and yet reasonably accurate turbulence model. The Baldwin-Lomax (B-L) turbulence model was evaluated for flow past a prolate spheroid at an angle of attack of $\alpha = 10°$ and a $Re = 4.2 \times 10^6$ [6]. The results show that comparison of surface pressure coefficient, C_p, with experiment was good up to $x/L = 0.689$ and deviated from the experimental results near the trailing edge. It is well known that the standard B-L model is inadequate for modeling separated flows. Hence, effort was focused on improving the turbulence models that are implemented in the incompressible flow solver. To this end, first, the Degani-Schiff (D-S) correction to the Baldwin-Lomax (B-L) algebraic turbulence model was incorporated. This involved selecting the first peak closer to the wall in the F distribution of the B-L model. Also, in the vicinity of the primary and secondary separated flow, the choice of F_{max} and y_{max} were such that the distribution of F and y are smooth in the streamwise and crossflow directions.

During the implementation of the D-S correction, it was found that in the original B-L implementation, in the calculation of y^+, the vorticity at the wall was used instead of the wall shear. Both these quantities are same for 2-D flows over flat plate but the differences could be large for surfaces with large curvature. Hence, the τ_w was computed and used in the y^+ calculation.

The B-L model with the τ_w correction and the B-L with D-S correction were employed first for the flow over a flat plate at $Re = 10^6$. All of these models yielded identical results. Next, these models were applied to compute the flow over a NACA0012 airfoil at an angle of attack $\alpha = 5°$ and

a $Re = 1.7 \times 10^5$. The C_p distribution on the airfoil in all the three cases, viz., B-L, B-L with τ_w correction and B-L with D-S correction, were compared to the experimental values. All the three cases are almost identical with the D-S correction exhibiting a marginally improving the results near the trailing edge. The comparison with the experiments, shown in Fig. 13, is good up to x/C of 0.5 and over predicts thereafter on both the suction and pressure sides.

Next, the Baldwin-Lomax (B-L) model with the Degani-Schiff (D-S) correction was applied to compute the flow past NACA 0012 airfoil at $Re = 1.7 \times 10^5$, and various angles of attack α from 5° to 11.5°. For the lower α case, both B-L and B-L with D-S correction yielded almost the same results. However, for the larger α case, the B-L with D-S correction was not converging adequately. A closer look at the μ_t distribution revealed that the distribution was not smooth near upper surface of the airfoil. When local time steps were employed to accelerate convergence, the μ_t distribution near the upper surface worsened and eventually degraded to μ_l. The cause for this behaviour was traced to the sensitivity in the selection of the first peak in the function F in the B-L model. In order to obtain the first peak near the wall, the D-S correction states that the peak in F is attained if F drops to 0.9 of its maximum value. If this factor was reduced, it was observed that the convergence improved. Finally, in order to obtain a fully converged solution, this search was turned off. The results were compared with experiments and also with B-L without the D-S correction. The C_p distribution on the upper surface was slightly improved with the D-S correction. The discrepancy near the trailing edge of the airfoil when compared to the experiments was still present. Another flow case with large separated flow was computed. This configuration is the flow past NACA 0012 at $Re = 3.9 \times 10^6$ and $\alpha = 20°$. Figure 14 shows that for this case, the B-L and B-L with D-S correction yielded similar results. The C_p distribution on the upper surface deviated from the experiments for almost 90% of the chord length. Therefore, it was clear that the D-S correction did not result in any improvement for the 2-D separated flows tested here.

8.4 UUV Hydrodynamics

First the flow over a 2-D Rankine oval near a wall was computed. The centerline of the 1:7 oval was situated at $0.75D$ from the wall, where D is the diameter of the oval. A grid consisting of approximately 350 points on the body and a total of 4600 points was employed. The computed force showed a suction force toward the wall, as expected. Figure 15 shows the surface pressure distribution and it is clear that a large suction pressure is created on the lower surface of the oval. Figures 16a and b show the grid that was employed and the pressure distribution for this configuration.

Next, the flow past a 3-D Rankine ovoid near a wall was computed. The centerline of the 1:7 ovoid was situated at $0.75D$ from the wall, where D is the diameter of the ovoid. A grid consisting of approximately 58K points and 325K tetrahedra was employed. A steady state solution was obtained in 500 steps. The residual errors dropped by more than 6 orders of magnitude with the use of local time steps. The C_p distribution on the symmetry plane of the body was obtained and is shown in Fig. 17. The distribution on the line closer to the wall was similar to that away from the wall. This is distinctly different from the 2-D result. The reason being that the flow has relief in the spanwise direction. The force towards the wall was approximately 0.0494, compared to the 2-D result of 6.0615. In order to establish grid independence, a finer grid was chosen. The computational domain in this case was halved using the fact that the flow is symmetric in the spanwise direction. The grid employed consisted of 94K points and 518K tetrahedra in the half

domain. The number of points on the body is almost 4 times that of the previous case. The C_p distribution on the symmetry plane was compared in Fig. 17. Results show grid convergence was achieved for the pressure distribution on the body. The computed force toward the wall was 0.0247 which is exactly half of the coarser grid result. Figure 18 shows the pressure contours on the surface of the ovoid, the symmetry plane and the wall.

Another series of computation of hydrodynamic forces and moments on the torpedo body at various lateral offsets from the submarine was performed. First in this series is the simulation where the centerline of the torpedo was located at $1D$ away from the hull, where D is the diameter of the torpedo. Both the submarine and the torpedo were moving at 5 knots. Three different grids were employed for the simulation; a coarse grid consisting of 37K points, a medium grid consisting of 89K points and a fine grid consisting of 148K points. Steady state solutions were obtained on these three grids. Forces and moments were computed and the results showed that the lateral force did not converge for the three cases. In order to locate the origin of this, the C_p distribution on the torpedo body along the symmetry plane was obtained. Figure 19a shows that the pressure distribution for the medium and fine grid results converged for most of the length of the torpedo. Considerable differences were seen near the propulsor plane of the torpedo as shown in Fig. 19b. This is due to the fact that both the propulsor inflow and outflow planes were blocked in our computation and recirculation zones formed in front of the inflow plane and behind the outflow plane. Hence, the force calculation was cut off before the propulsor plane, and a grid converged result was obtained.

The results of the simulation where the centerline of the torpedo was located at $1D$ away from the hull, where D is the diameter of the torpedo, showed that a grid consisting of 89K points was sufficient to obtain a grid converged solution. Figure 20 shows the pressure distribution on the torpedo and the submarine hull. This grid was employed to simulate the flow with the centerline of the torpedo $0.625D$ and $0.75D$ away from the submarine hull. The computation of the force on the torpedo was cut off before the propulsor plane. For the case of $H/D = 1D$, the lateral force on the torpedo was $8.94lbs$; for $H/D = 0.75D$, the force was $14.59lbs$ and for $H/D = 0.635D$, the force increased to $20.76lbs$. The C_p distribution on the symmetry plane of the torpedo, $y = 0$, is shown in Fig. 21. It is clear that variation of H/D has very little effect on the surface away from the hull and has a pronounced effect on the inner surface towards the submarine hull. The post-processing code was also modified to extract the surface pressure distribution on the hull along the symmetry plane. Figure 22 shows that for all the three cases, the C_p attained a positive maximum ahead of the nose of the torpedo and drops to a minimum and recovers to the free stream value. The magnitude of the maximum and minimum peaks increase with decrease in H/D, as expected.

9. SUMMARY AND CONCLUSIONS

A new, 3-D incompressible flow solver based on simple finite elements with adaptive remeshing has been developed. The main algorithmic ingredients for mesh movement were described. These were directed towards reducing the number of remeshings and hence the total CPU time involved in the simulation. The torpedo launch problem represents just one of many cases where the combination of adaptive remeshing techniques, flow solvers for transient problems with moving grids, and integrators for rigid body motion allows the simulation of fully coupled fluid-rigid body interaction problems of arbitrary geometric complexity in three dimensions. Areas that deserve further study are the diffusive effect of interpolation while remeshing and extension to Navier-Stokes problems.

Acknowledgments

This work was supported by the Laboratory for Computational Physics and Fluid Dynamics of the Naval Research Laboratory and the Naval Undersea Warfare Center. The authors wish to thank John Schwemin and Steve Jordan of NUWC, for their assistance and support throughout the course of this work. Computing support was provided by Numerical Aerodynamic Simulation (NAS) Program and DoD HPC Program and is gratefully acknowledged.

10. REFERENCES

1. Schwemin, J.A., Sowle, J.M., McNally, G.A. and Jordan, S.A., "Launchway Technology," *NUWC-NPT TM* 9321136, November 1993.
2. Löhner, R., "Matching Semi-Structured and Unstructured Grids for Navier-Stokes Calculations," *AIAA*-93-3348, CP-933, Proceedings of the 11th AIAA CFD conference, Orlando, FL, pp. 555-564, July 1993.
3. Gregoire, J.P., Benque, J.P., Lasbleiz, P. and Goussebaile, J., "3-D Industrial Flow Calculations by Finite Element Method," *Springer Lecture Notes in Physics* **218**, pp. 245-249, 1985.
4. Martin, D. and Löhner, R., "An Implicit Linelet-Based Solver for Incompressible Flows," *AIAA*-92-0668, Washington, D.C., 1992.
5. Ramamurti, R. and Löhner, R., "Evaluation of an Incompressible Flow Solver Based on Simple Elements," *Advances in Finite Element Analysis in Fluid Dynamics*, FED Vol. 137, Editors: M.N. Dhaubhadel et al., ASME Publication, New York, pp. 33-42, 1992.
6. Ramamurti, R., Löhner, R. and Sandberg, W.C., "Evaluation of a Scalable 3-D Incompressible Finite Element Solver," *AIAA*-94-0756, Washington, D.C., 1994.
7. Sandberg, W.C., "The Estimation of Ship Motion Induced Loads," 4th Ship Technology and Research Symposium, *SNAME*, pp. 347-351, 1979.
8. Löhner, R. and Parikh, P., "Three-Dimensional Grid Generation by the Advancing Front Method," *Int. J. Num. Meth. Fluids* **8**, pp. 1135-1149, 1988.
9. Peraire, J., Morgan, K. and Peiro, J., " Unstructured Finite Element Mesh Generation and Adaptive Procedures for CFD," *AGARD*-CP-464, **18**, 1990.
10. Löhner, R., "Some Useful Data Structures for the Generation of Unstructured Grids," *Comm. Appl. Num. Meth.* **4**, pp. 123-135, 1988.
11. Löhner, R., "An Adaptive Finite Element Scheme for Transient Problems in CFD," *Comp. Meth. Appl. Mech. Eng.* **61**, pp. 323-338, 1987.
12. Boschitsch, A.H. and Quackenbush, T.R., "High Accuracy Computations of Fluid-Structure Interaction in Transonic Cascades," *AIAA*-93-0485, 1993.
13. Batina, J.T., "Unsteady Euler Airfoil Solutions Using Unstructured Dynamic Meshes," *AIAA J.*, **8**, pp. 1381-1388, 1990.
14. Rausch, R.D., Batina, J.T. and Yang, H.T.Y., "Three-Dimensional Time-Marching Aerolastic Analyses Using an Unstructured-Grid Euler Method," *AIAA J.* **31**, 9, pp. 1626-1633, 1993.
15. Löhner, R., "An Adaptive Finite Element Solver for Transient Problems with Moving Bodies," *Comp. Struct.* **30**, pp. 303-317, 1988.

16. Löhner, R., "Three-Dimensional Fluid-Structure Interaction Using a Finite Element Solver and Adaptive Remeshing," *Computer Systems in Engineering* **1**, 2-4, pp. 257-272, 1990.

17. Löhner, R. and Baum, J.D., "Three-Dimensional Store Separation Using a Finite Element Solver and Adaptive Remeshing," *AIAA*-91-0602, 1991.

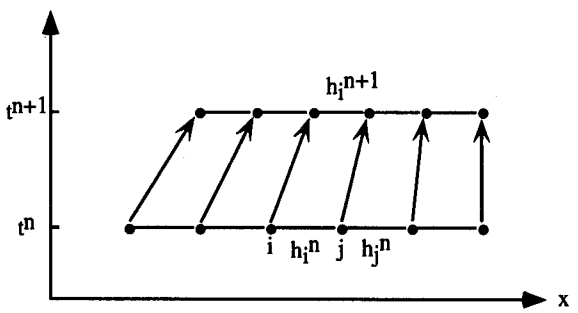


Fig. 1. Smoothing of Velocity Field

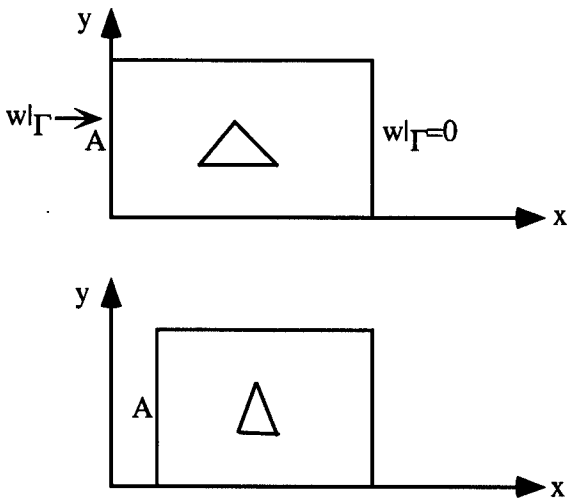


Fig. 2. Smoothing of Coordinates

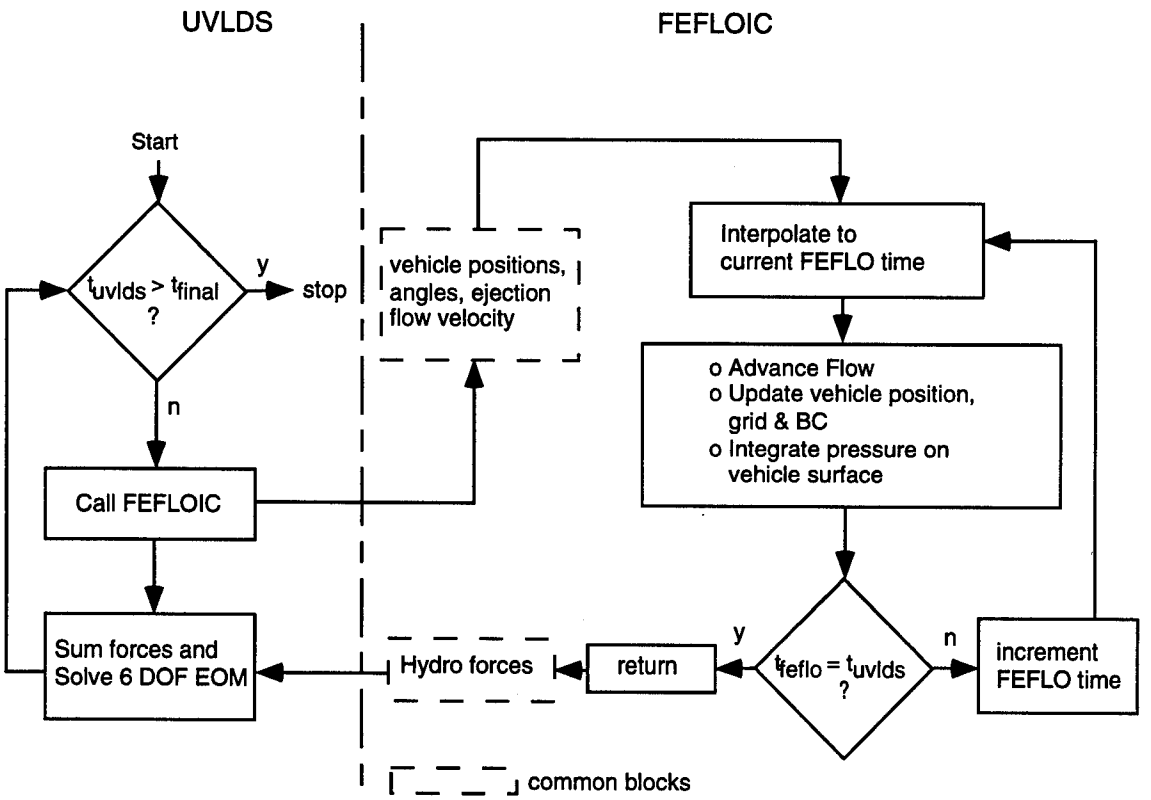


Fig. 3. Schematic of the Hybrid Simulation

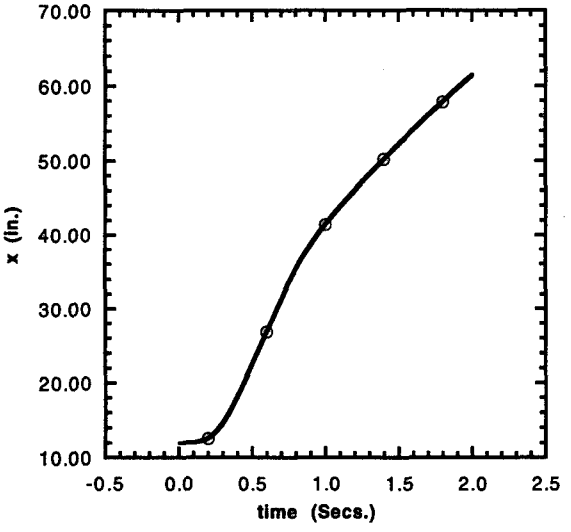


Fig. 4. Prescribed Torpedo Trajectory

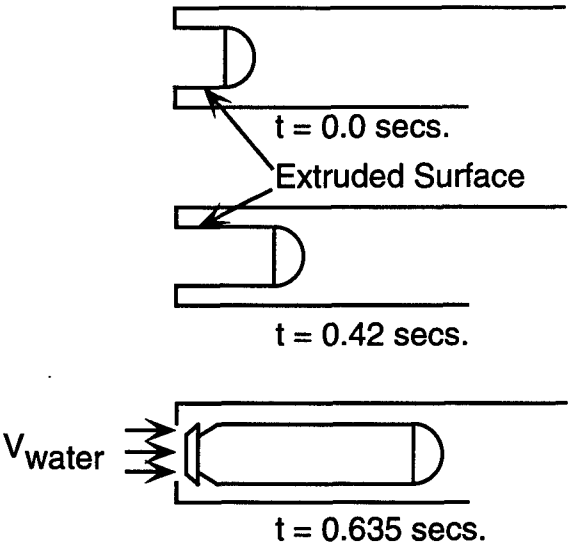


Fig. 5. Extrusion Process

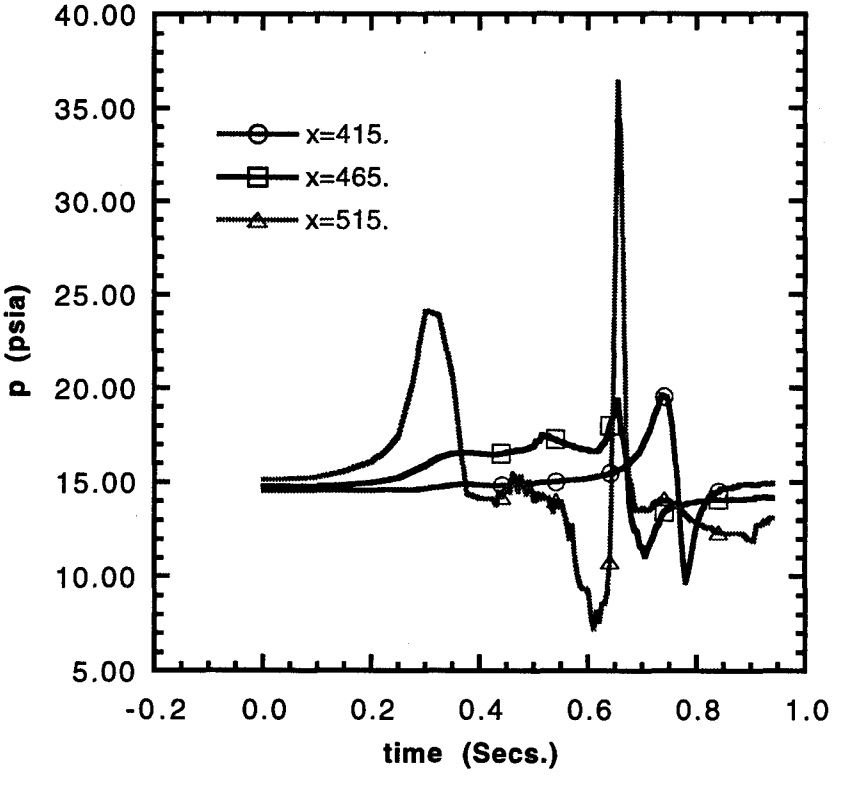


Fig. 6. Time History of Pressure

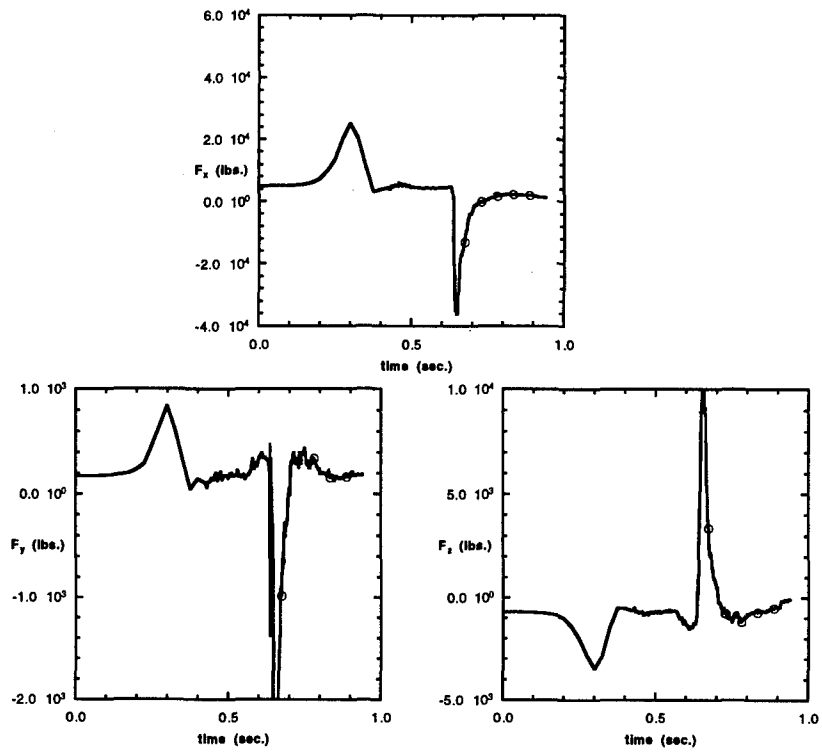
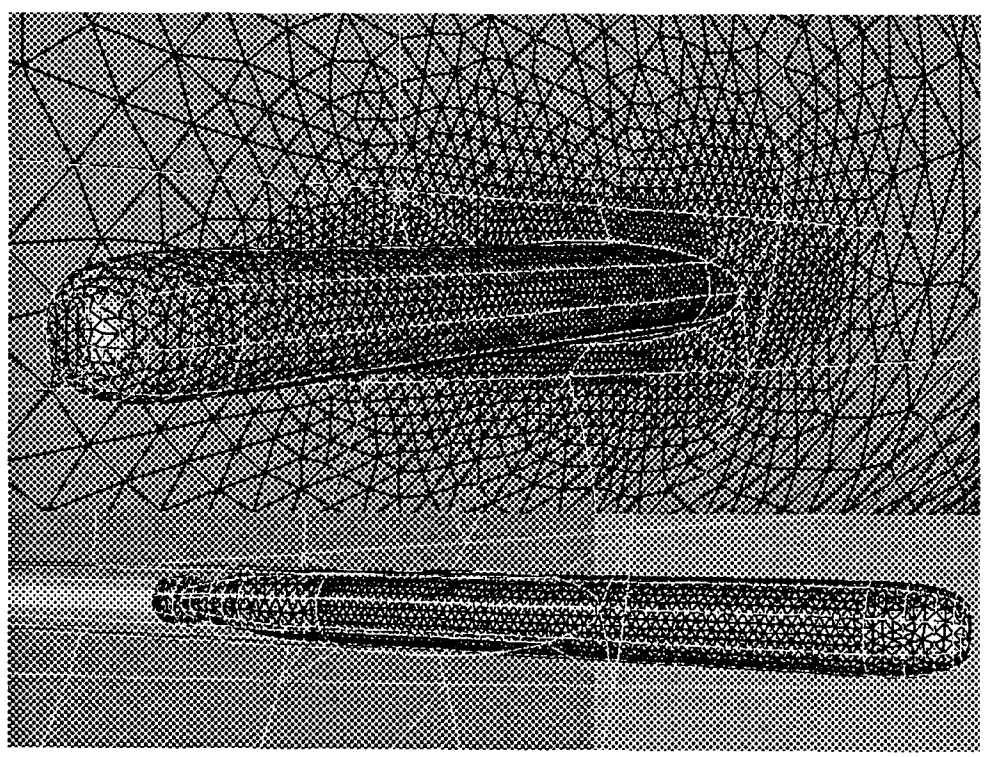
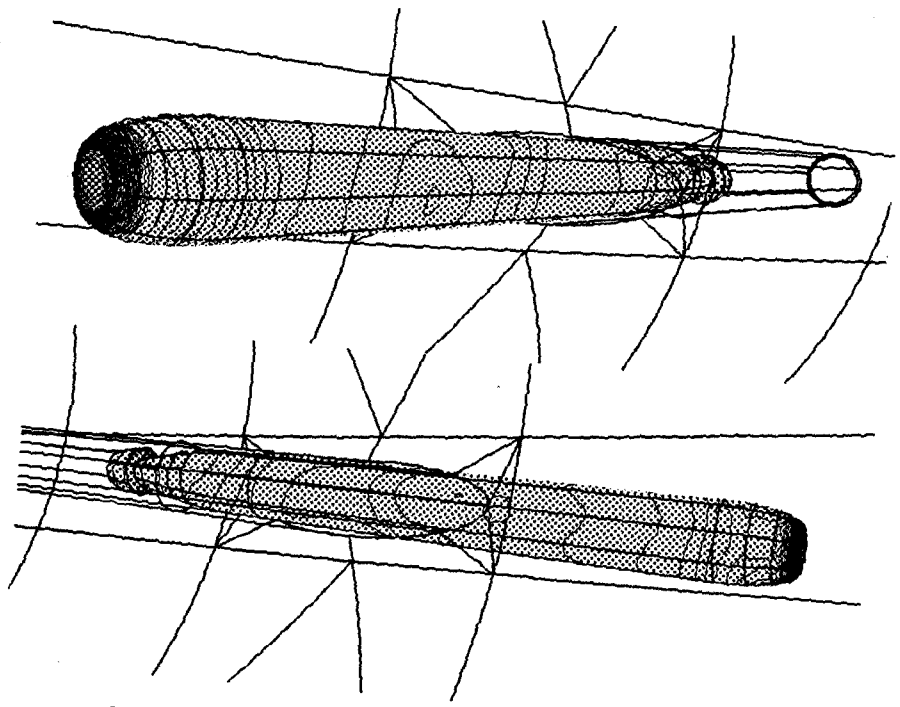


Fig. 7. Time History of Force on the Torpedo

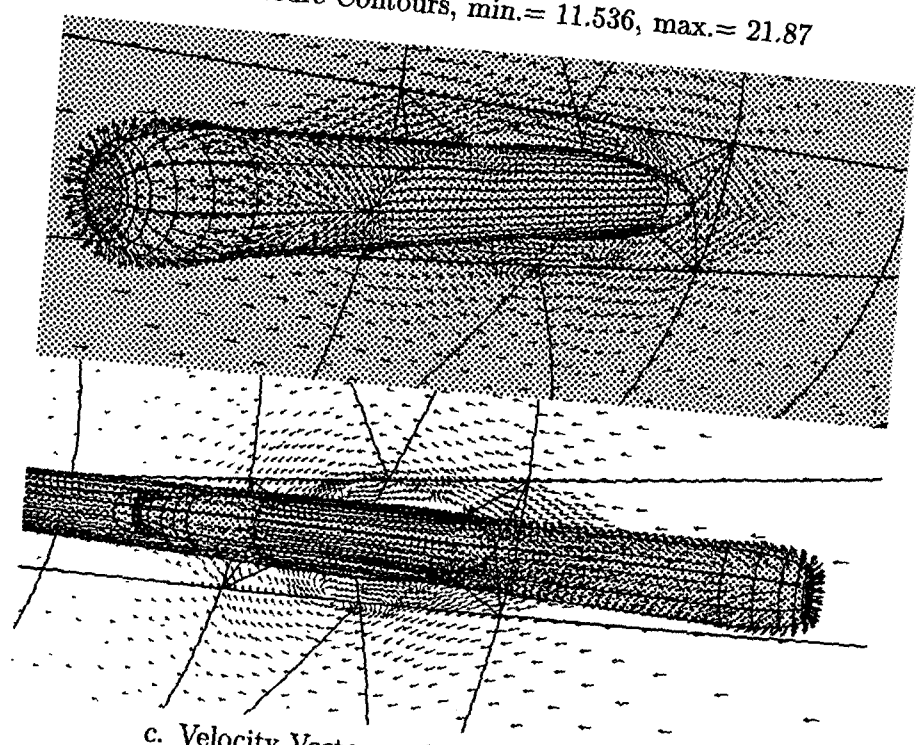


a. Surface Mesh, nelem= 347,176, npoin= 64,357

Fig. 8 – Simulation of a Torpedo Launch, time= 0.94 secs



b. Pressure Contours, min.= 11.536, max.= 21.87



c. Velocity Vectors, min = 0., max. = 390.86

Fig. 8. Simulation of a Torpedo Launch, time= 0.94 secs

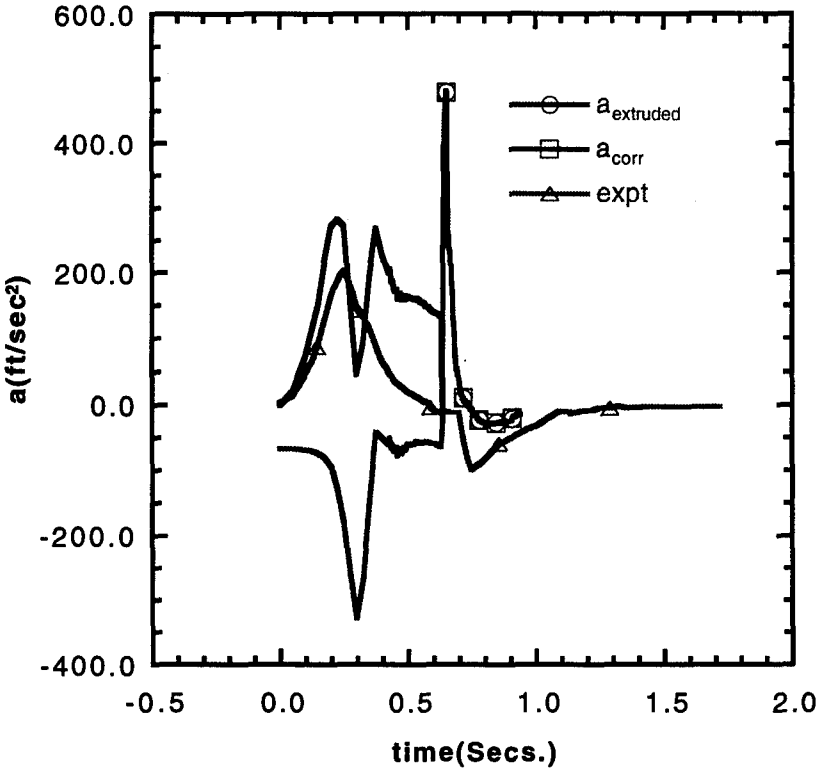


Fig. 9. Comparison of Centerline Acceleration

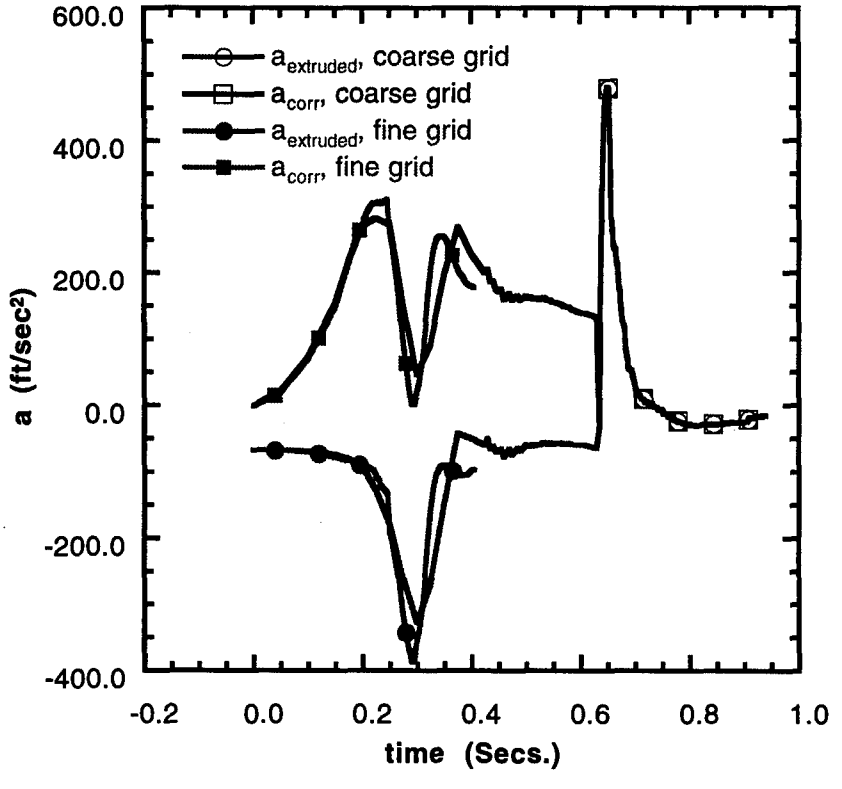


Fig. 10. Effect of Grid Refinement

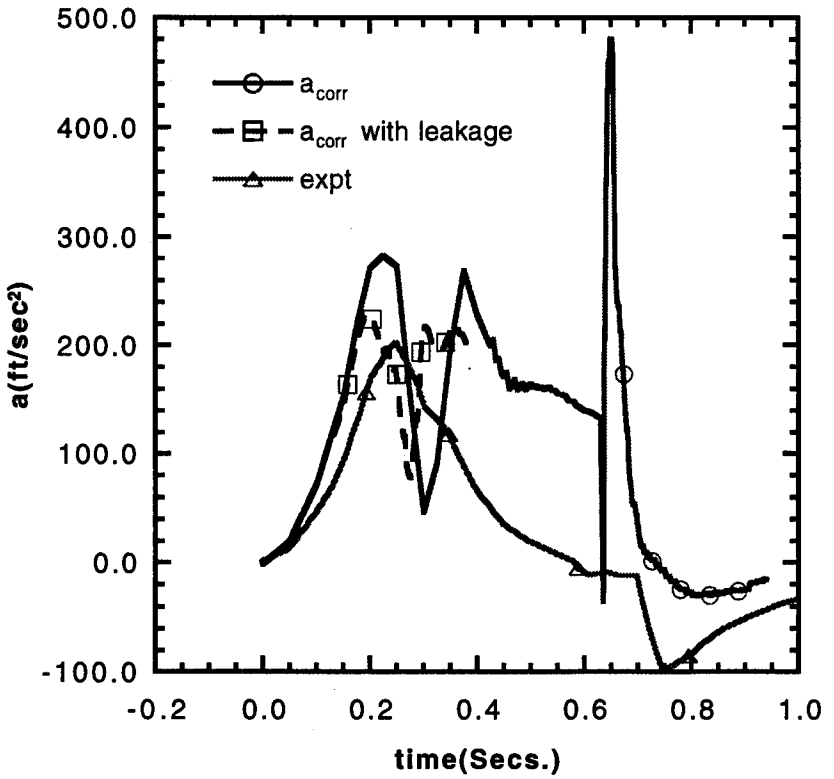


Fig. 11. Effect of Water Leakage Base of the Launchway

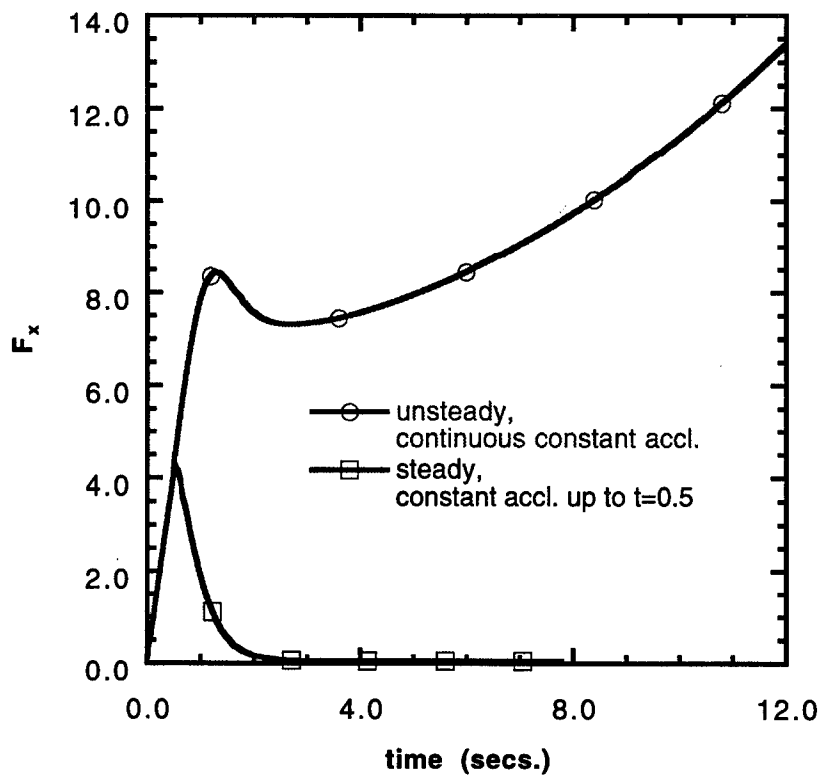


Fig. 12. Force due to Added Mass

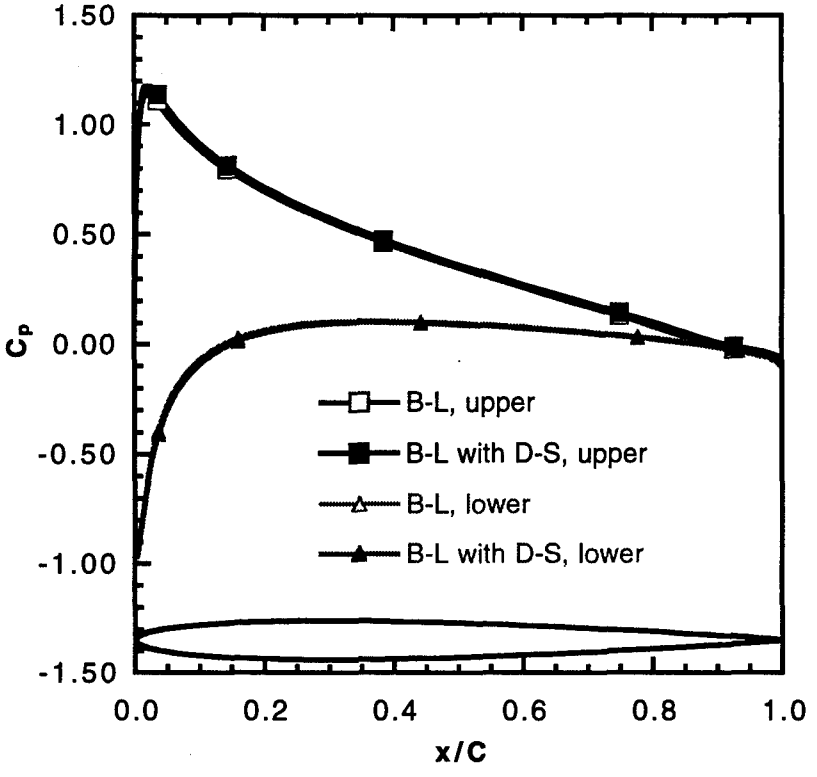


Fig. 13. Surface Pressure Distribution, $Re = 1.7 \times 10^5, \alpha = 5°$

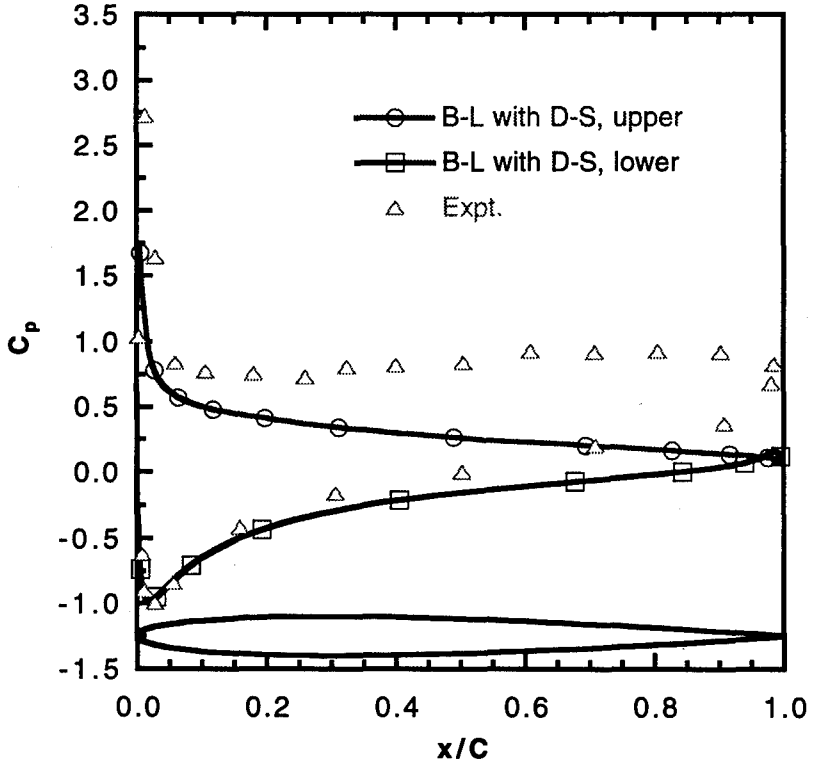


Fig. 14. Surface Pressure Distribution, $Re = 3.9 \times 10^6, \alpha = 20°$

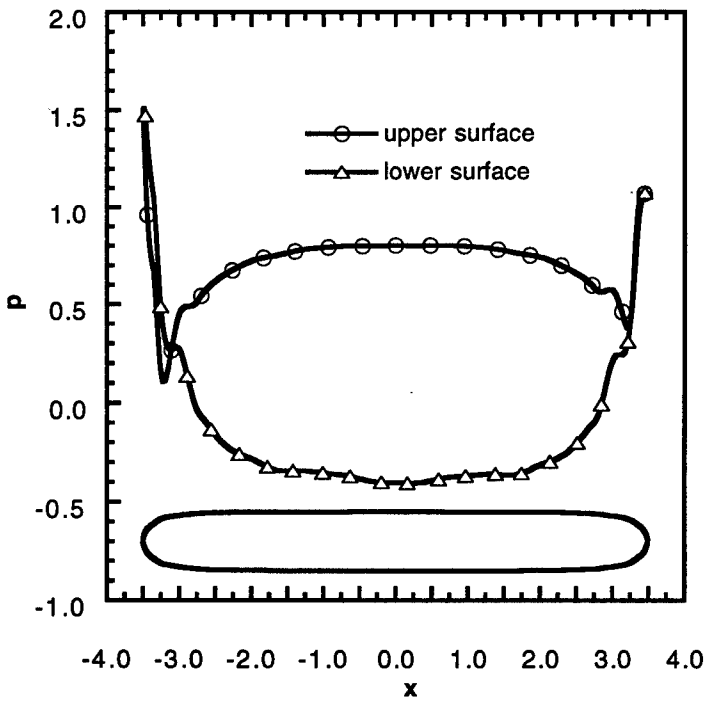
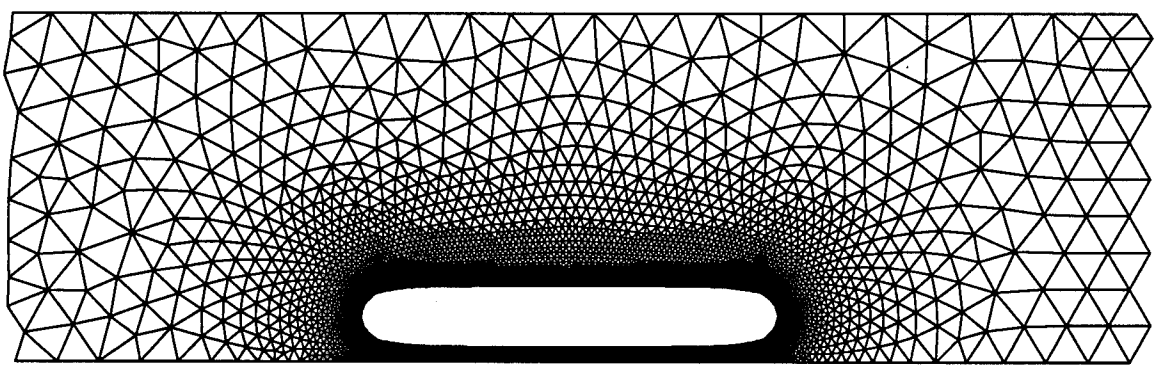
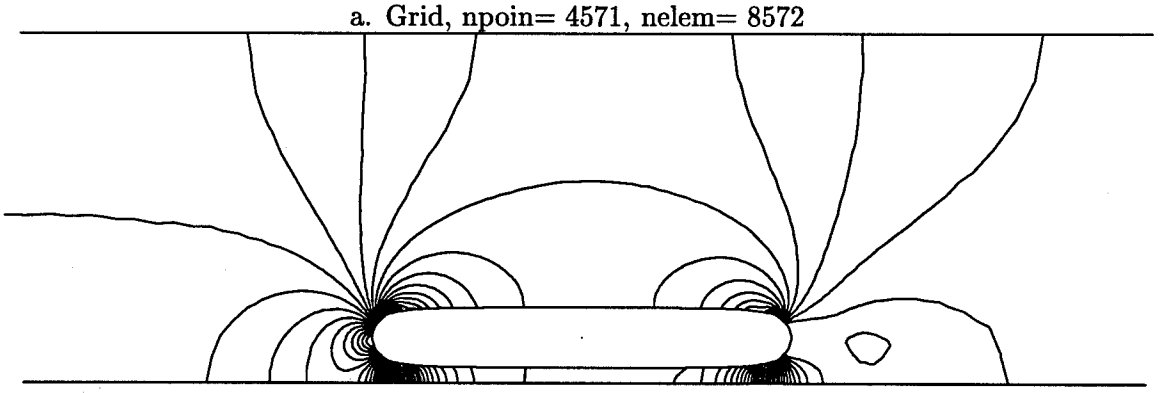


Fig. 15. Surface Pressure Distribution on 1:7 Rankine Oval



a. Grid, npoin= 4571, nelem= 8572



b. Pressure Contours, min= -0.4, max=1.51, Δp= 0.0478

Fig. 16. Flow Past a 1:7 Rankine Oval

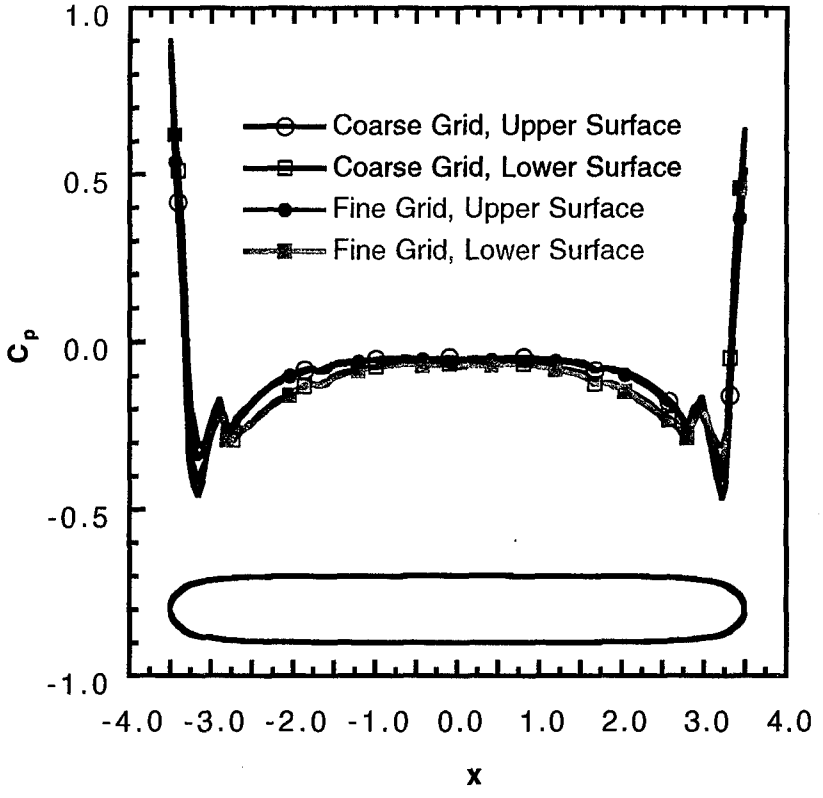


Fig. 17. Surface Pressure Distribution on the Symmetry Plane for 1:7 Rankine Ovoid

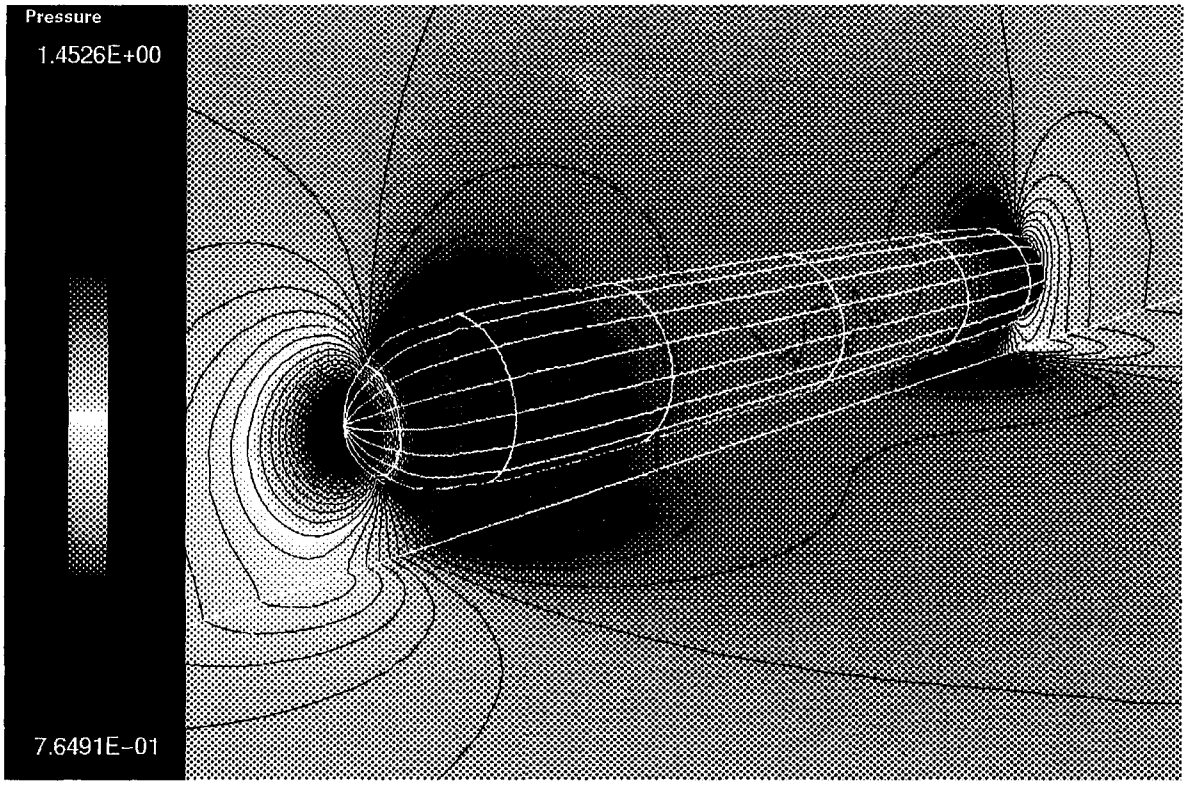
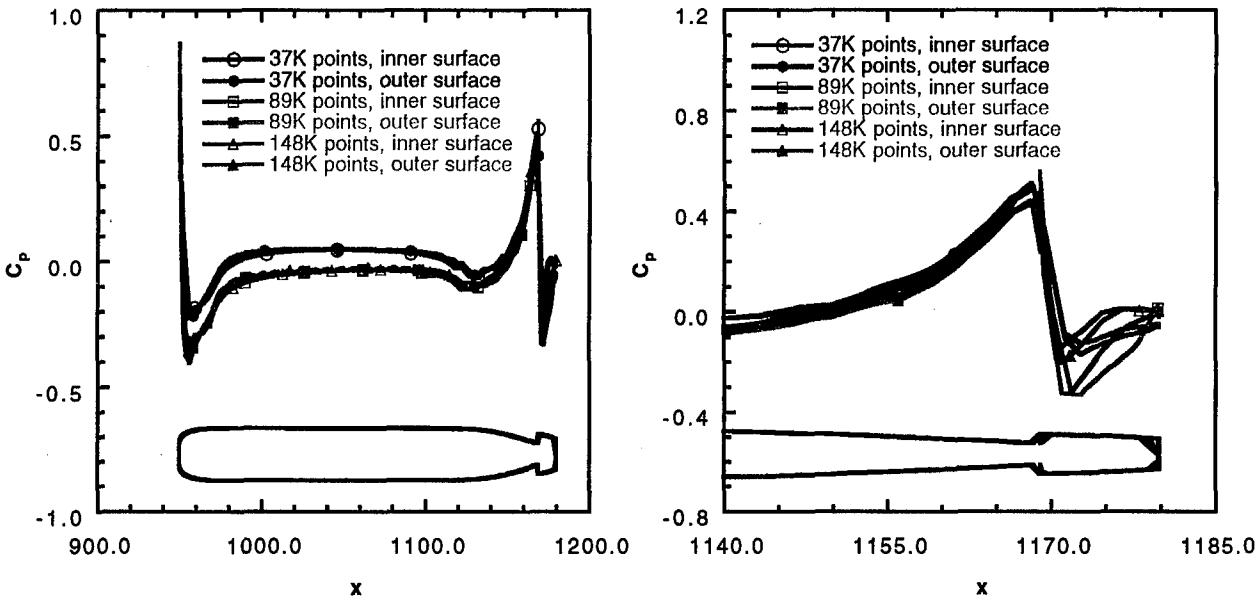


Fig. 18. Pressure Distribution for Flow Past a 1:7 Rankine Ovoid



a. Entire Torpedo b. Propulsor Region

Fig. 19. Effect of Grid Refinement on Surface Pressure Distribution for Flow Past a Torpedo in the Vicinity of a Submarine, H/D=1.0

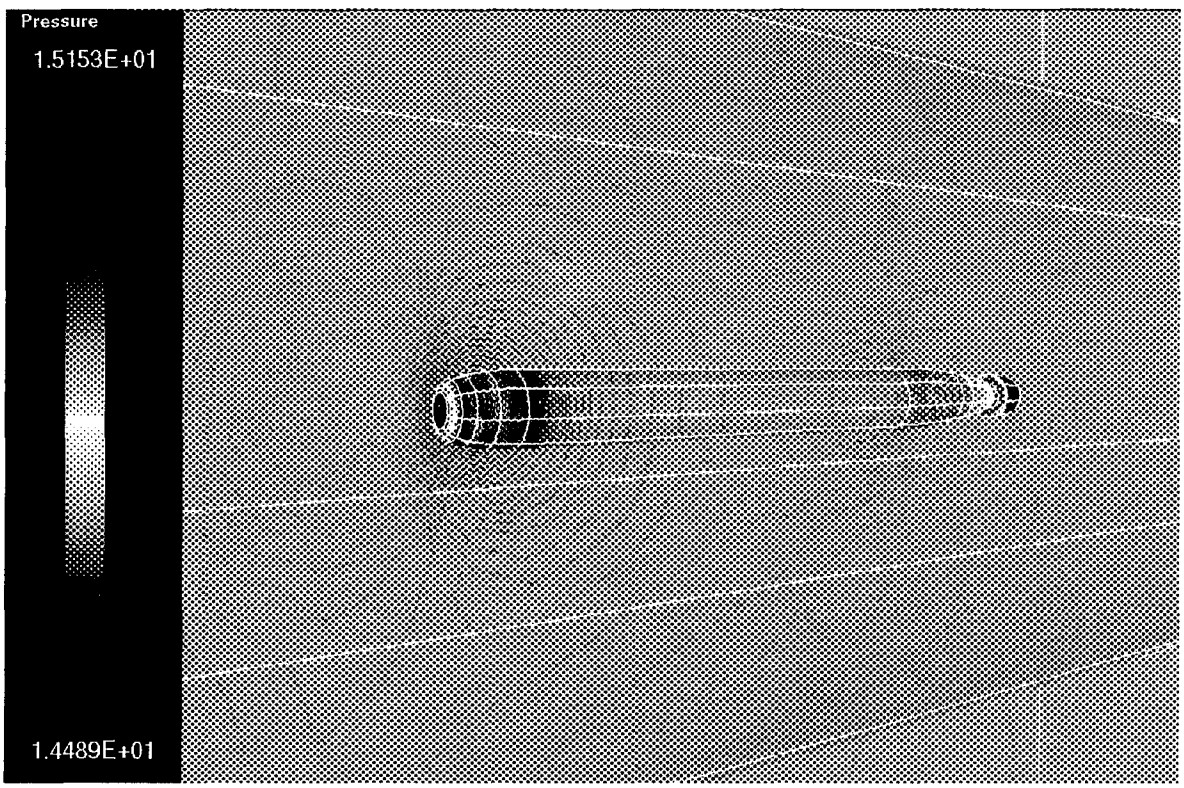


Fig. 20. Flow Past a Torpedo in the Vicinity of a Submarine, H/D=1.0, Pressure Contours

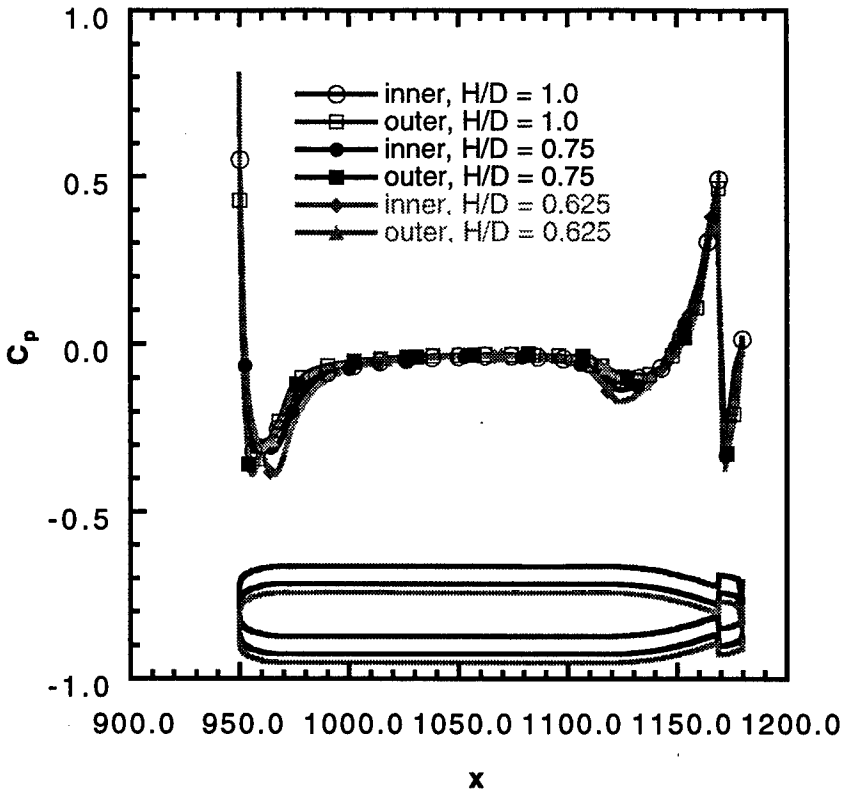


Fig. 21. Effect of Lateral Offset on Torpedo Surface Pressure

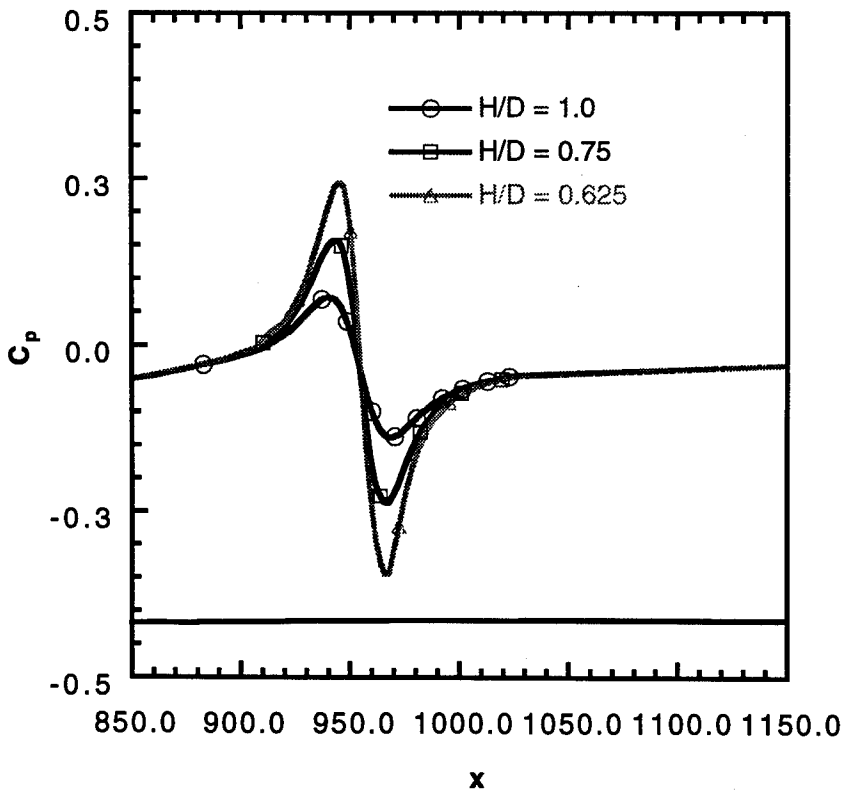


Fig. 22. Surface Pressure Distribution on the Submarine Wall